**LAPORAN PRAKTIKUM 3**

**Analisis Algoritma**

**(ANALGO)**



**Muhamad Farid Ridho Rambe**

**140810180033**

**Kelas A**

**Program Studi S-1 Teknik Informatika Departemen Ilmu Komputer Fakultas Matematika dan Ilmu Pengetahuan Alam Universitas Padjadjaran**

Pendahuluan

Minggu lalu kita sudah mempelajari menghitung kompleksitas waktu T(n) untuk semua operasi yang ada pada suatu algoritma. Idealnya, kita memang harus menghitung semua operasi tersebut. Namun, untuk alasan praktis, kita cukup menghitung operasi abstrak yang **mendasari suatu** **algoritma**, dan memisahkan analisisnya dari implementasi. Contoh pada algoritma searching,operasi abstrak yang mendasarinya adalah operasi perbandingan elemen x dengan elemen-elemen dalamlarik.Denganmenghitungberapaperbandinganuntuktiap-tiapelemennilainsehinggakita dapatmemperoleh **efisiensi relative** darialgoritmatersebut. Setelahmengetahui T(n) kitadapat menentukan **kompleksitas waktu asimptotik** yang dinyatakan dalam notasi Big-O, Big-Ω, Big-Θ, dan little-ω.

Setelah mengenal macam-macam kompleksitas waktu algoritma (best case, worst case, dan average case), dalam analisis algoritma kita selalu mengutamakan perhitungan **worst case** dengan alasan sebagaiberikut:

* Worst-case running time merupakan *upper bound* (batas atas) dari running time untuk input apapun. Hal ini memberikan jaminan bahwa algoritma yang kita jalankan tidak akan lebih lama lagi dari ***worst-case***
* Untuk beberapa algoritma, ***worst-case*** cukup sering terjadi. Dalam beberapa aplikasi pencarian, pencarian info yang tidak ada mungkin sering dilakukan.
* Pada kasus ***average-case*** umumnya lebih sering seperti ***worst-case.*** *Contoh:* misalkan kita secara randommemilihn angka dan mengimplementasikan insertionsort, ***average-case=*** ***worst-case*** yaitu fungsi kuadratik darin.

Perhitunganworstcase(*upperbound*)dalam kompleksitaswaktuasimptotikdapatmenggunakan **Big-O Notation. Perhatikan pembentukan Big-O Notation berikut!**

**Misalkan kita memiliki kompleksitas waktu T(n) dari sebuah algoritma sebagai berikut:**

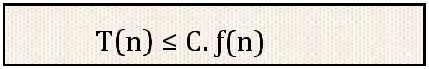
T(n) = 2n2 + 6n + 1

* Untuk n yang besar, pertumbuhan T(n) sebanding dengan n2
* Suku 6n + 1 tidakberartijikadibandingkandengan 2n2, danbolehdiabaikansehingga T(n) = 2n2 + suku-suku lainnya.
* Koefisien 2 pada 2n2boleh diabaikan, sehingga T(n) = O(n2)  **Kompleksitas Waktu**

**Asimptotik**

**DEFINISI BIG-O NOTATION**

**Definisi 1.** T(n)= 0(ƒ(n)) ***artinya*** T(n)berorde paling besarƒ(n)bila terdapat konstanta*C*dannosedemikian sehingga



Untuk n ≥ no

Jika n dibuat semakin besar, waktu yang dibutuhkan tidak akan melebihi konstanta C dikalikan dengan ƒ(n),  ƒ(n) adalah *upper bound*.

Dalam proses **pembuktian Big-O**, perlu dicari nilai no dan nilai C sedemikan sehingga terpenuhi kondisi T(n) ≤ C.ƒ(n).

**Contoh soal 1:**

Tunjukan bahwa, T(n) = 2n2 + 6n + 1 = 0(n2)

**Penyelesaian:**

Kita mengamati bahwa n ≥ 1, maka n ≤ n2 dan 1 ≤ n2 sehingga

2n2 + 6n + 1 ≤ 2n2 + 6n2 + n2 = 9n2, untuk n ≥ 1

Maka kita bisa mengambil C=9 dan nO=1 untuk memperlihatkan:

T(n) = 2n2 + 6n + 1 = 0(n2)

**BIG-O NOTATION DARI POLINOMIAL BERDERAJAT M**

Big-O Notation juga dapat ditentukan dari Polinomial n berderajat m, dengan TEOREMA 1 sebagai berikut:

Polinomial n berderajat N dapat digunakan untuk memperkirakan kompleksitas waktu asimptotik dengan mengabaikan suku berorde rendah

Contoh: T(n) = 3n3 + 6n2 + n + 8 = O(n3)**,** dinyatakan pada



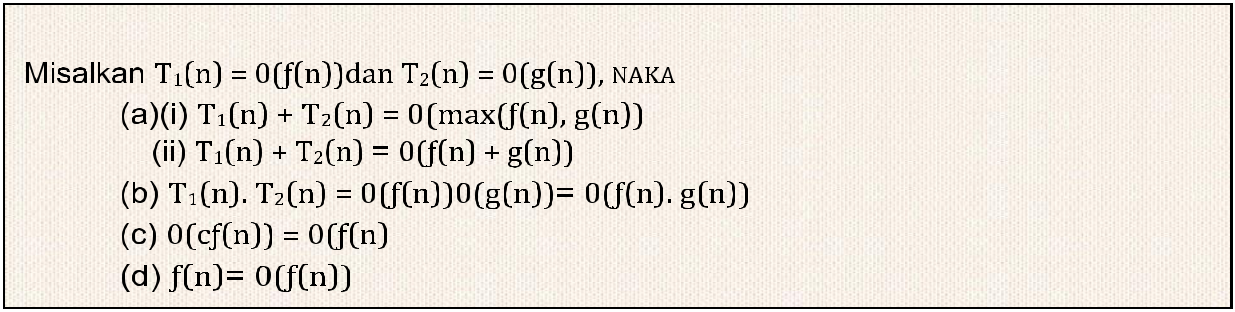
**TEOREMA 1**

Artinya kita mengambil suku paling tinggi derajatnya (“Mendominasi”) yang diartikan laju pertumbuhannya lebih cepat dibandingkan yang lainnya ketika diberikan sembarang besaran input. Besaran dominan lainnya adalah:

* Eksponensial mendominasi sembarang perpangkatan (yaitu, yn > np, y > 1)
* Perpangkatan mendominasi ln n (yaitu np > ln n)
* Semua logaritma tumbuh pada laju yang sama (yaitu a log(n) = b log(n)
* n log n tumbuh lebih cepat daripada n tetapi lebih lambat dari n2

Teorema lain dari Big-O Notation yang harus dihafalkan untuk membantu kita menentukan nilai

Big-O dari suatu algoritma adalah:



**TEOREMA 2**

Berikut adalah contoh soal yang mengaplikasikan Teorema 2 dari Big-O notation:

**Contoh Soal 2**

Misalkan, T1(n) = 0(n)dan T2(n) = 0(n2), dan T3(n) = 0(NN), dengan m sebagai peubah, maka

|  |  |  |
| --- | --- | --- |
| (a) | T1(n) + T2(n) = 0(max(ƒ(n, n2) = 0(n2) | Teorema 2(a)(i) |
| (b) T2(n) + T3(n) = 0(n2 + NN) | | Teorema 2(a)(ii) |
| (c) | T1(n). T2(n) = 0(n. n2 )= 0(n3) | Teorema 2(B) |

|  |  |  |
| --- | --- | --- |
| **Contoh Soal 3** | |  |
| (d) | 0(5n2 )= 0(n2) | Teorema 2(C) |
| (e) | n2 = 0(n2) | Teorema 2(D) |

**Aturan Menentukan Kompleksitas Waktu Asimptotik**

* Cara I

Jika kompleksitas waktu T(n) dari algoritma sudah dihitung, maka kompleksitas waktu asimptotiknya dapat langsung ditentukan dengan mengambil suku yang mendominasi fungsi T dan menghilangkan koefisiennya (sesuai TEOREMA 1)

**Contoh:**

Pada algoritma cariMax, T(n) = n − 1 = 0(n)

* Cara 2

Kita bisa langsung menggunakan notasi Big-O, dengan cara:

Pengisian nilai (assignment), perbandingan, operasi aritmatika (+,-,/,\*, div, mod), read, write, pengaksesan elemen larik, memilih field tertentu dari sebuah record, dan pemanggilan function/void membutuhkan waktu O(1)

**Contoh Soal 4:**

Tinjau potongan algoritma berikut:

|  |  |
| --- | --- |
| read(x) | O(1) |
| x  x + 1 | O(1) +O(1)=O(1) |
| write(x) | O(1) |

Kompleksitas waktu asimptotik algoritmanya 0(1) + 0(1) + 0(1) = 0(1)

|  |  |  |
| --- | --- | --- |
| Penjelasan: |  |  |
| 0(1) + 0(1) + 0(1) = 0(NAX(1,1)) + 0(1) | | Teorema 2(A)(i) |
| = 0(1) + 0(1) | |  |
|  | = 0(NAX(1,1)) | Teorema 2(A)(ii) |
| = 0(1) | |  |

**DEFINISI BIG-Ω DAN BIG-Θ NOTATION**

Notasi Big-O hanya menyediakan batas atas *(upper bound)* untuk perhitungan kompleksitas waktu asimptotik, tetapi tidak menyediakan batas bawah *(lower bound)*. Untuk itu, lower bound dapat ditentukan dengan Big-Ω Notation dan Big-θ Notation.

**Definisi Big-Ω Notation:**

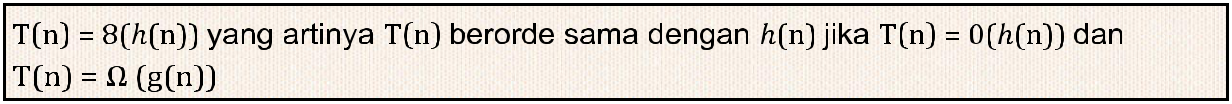
T(n) = Ω(g(n)) yang artinya T(n)berorde paling kecil g(n) bila terdapat konstanta C dan nosedemikian sehingga



T(n) ≥ C.(g(n))

untuk n ≥ no

**Definisi Big-θ Notation:**



**Contoh Soal 5:**

Tentukan Big-Ω dan Big- Θ Notation untuk T(n) = 2n2 + 6n + 1

Penyelesaian:

Karena 2n2 + 6n + 1 ≥ 2n2 untuk n ≥ 1, dengan mengambil C=2, kita memperoleh 2n2 + 6n + 1 = G(n2)

Karena 2n2 + 6n + 1 = 0(n2) dan 2n2 + 6n + 1 = G(n2), maka 2n2 + 6n + 1= 8(n2) **Penentuan Big-Ω dan Big-** © **dari Polinomial Berderajat m**

Sebuah fakta yang berguna dalam menentukan orde kompleksitas adalah dari suku tertinggi di dalam polinomial berdasarkan teorema berikut:



**TEOREMA 3**

**Contoh soal 6:**

Bila T(n) = 6n4 + 12n3 + 24n + 2,

maka T(n) adalah berorde n4, yaitu 0(n4), Ω(n4), dan Θ(n4).

Latihan Analisa

Minggu ini kegiatan praktikum difokuskan pada latihan menganalisa, sebagian besar tidak perlu menggunakan komputer dan mengkoding program, gunakan pensil dan kertas untuk menjawab persoalan berikut!

1. Untuk T(n) = 2 + 4 + 8 + 16 + ⋯ + 2n, tentukan nilai C, f(n), no, dan notasi Big-O sedemikian sehingga T(n) = 0(ƒ(n)) jika T(n) ≤ C untuk semua n ≥ nO
2. Buktikan bahwa untuk konstanta-konstanta positif p, q, dan r:

T(n) = en2 + qn + r adalah 0(n2), Ω(n2), dan Θ(n2)

3. Tentukanwaktukompleksitasasimptotik(Big-O,Big-Ω, danBig-Θ)darikodeprogramberikut:

for k  1 to n do

for i  1 to n do

for j  to n do

wij  wij or wik and wkj

endfor

endfor

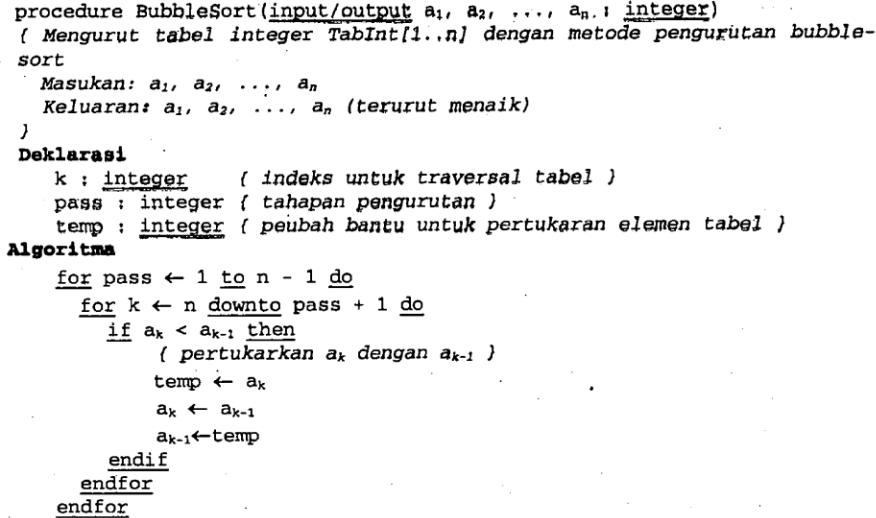
endfor

1. Tulislah algoritma untuk menjumlahkan dua buah matriks yang masing-masing berukuran n x n. Berapa kompleksitas waktunya T(n)? dan berapa kompleksitas waktu asimptotiknya yang dinyatakan dalam Big-O, Big-Ω, dan Big-Θ?

’

1. Tulislahalgoritmauntukmenyalin(copy)isisebuahlarikkelariklain.Ukuranelemenlarikadalah n elemen. Berapa kompleksitas waktunya T(n)? dan berapa kompleksitas waktu asimptotiknya yang dinyatakan dalam Big-O, Big-Ω, dan Big-Θ?

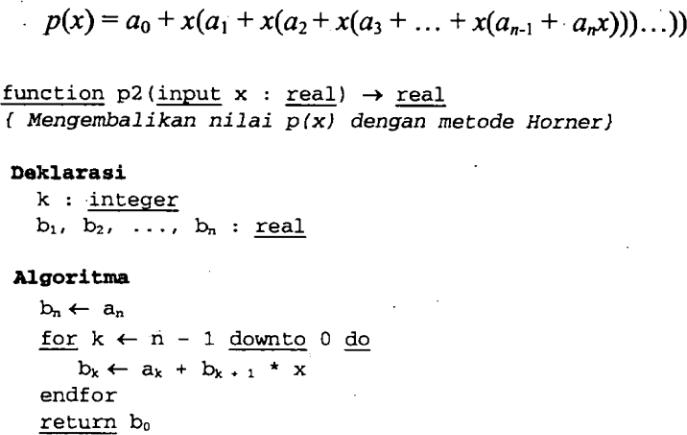
6. Diberikan algoritma Bubble Sort sebagai berikut:



1. Hitung berapa jumlah operasi perbandingan elemen-elemen tabel!
2. Berapa kali maksimum pertukaran elemen-elemen tabel dilakukan?
3. Hitungkompleksitaswaktuasimptotik(Big-O,Big-Ω, dan Big-Θ)darialgoritma BubbleSort tersebut!
4. Untuk menyelesaikan problem X dengan ukuran N tersedia 3 macam algoritma:
   1. Algoritma A mempunyai kompleksitas waktu O(log N)
   2. Algoritma B mempunyai kompleksitas waktu O(N log N)
   3. Algoritma C mempunyai kompleksitas waktu O(N2)

Untuk problem X dengan ukuran N=8, algoritma manakah yang paling cepat? Secara asimptotik, algoritma manakah yang paling cepat?

8. Algoritma mengevaluasi polinom yang lebih baik dapat dibuat dengan metode Hornerberikut:



Hitunglah berapa operasi perkalian dan penjumlahan yang dilakukan oleh algoritma diatas, Jumlahkan kedua hitungan tersebut, lalu tentukan kompleksitas waktu asimptotik (Big-O)nya. Manakah yang terbaik, algoritma p atau p2

Jawab :

A document with black text

Description automatically generatedA close up of text on a white background

Description automatically generatedA close up of text on a white background

Description automatically generatedA close up of text on a white background

Description automatically generated